Data-driven Challenges to Architectures and Systems: An Architecture Perspective

Peter M. Kogge
Assoc. Dean of Engineering
University of Notre Dame
IBM Fellow (retired)

My 4 Questions

1. "On the spot" question. Can it be addressed by current architectures & systems? If so, how? If not, what advances are needed to solve the issue?
2. What are architecture/systems issues most applications people overlook?
3. What, in my current area of research, is the most exciting development which will benefit applications the most?
4. Looking back on my experience and the architecture/system decisions I have taken, what would I have done differently?

Q1: On the Spot Question

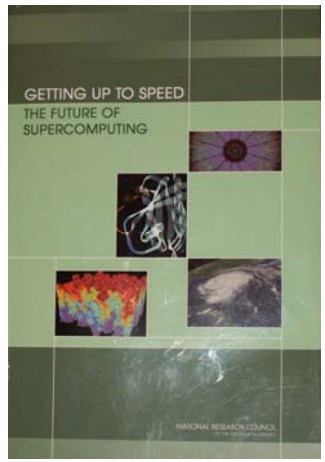
- **Choose a Data Driven Challenge from prior speakers.**
- **Can it be addressed by current architectures & systems?**
- **If so, how?**
- **If not, what advances are needed to solve the issue**

Q2: Issues Most Overlooked = Memory *& Getting There*

Also Data Mining, Large Dynamic Graphs

	Stockpile	Intelligence	Defence	Climate	Plasma	Transportation	Bio-info	Health&Safety	Earthquakes	Geophysics	Astrophysics	Materials	Organ. Systems
Performance Flops			1	X	X						X		
Memory Capacity		X				3	2					X	
Memory Bandwidth		X		X							X	X	4
Memory Latency	X	X		X							X		4
Interconnect Bandwidth		X		X							X	X	4
Interconnect Latency	X	X		X							X		4

1 Radar Cross section
2 Genomics
3 Automobile Noise
4 Biological Systems Modeling



(from NRC's "Getting Up to Speed")

We Talk About the Memory Wall, But ….

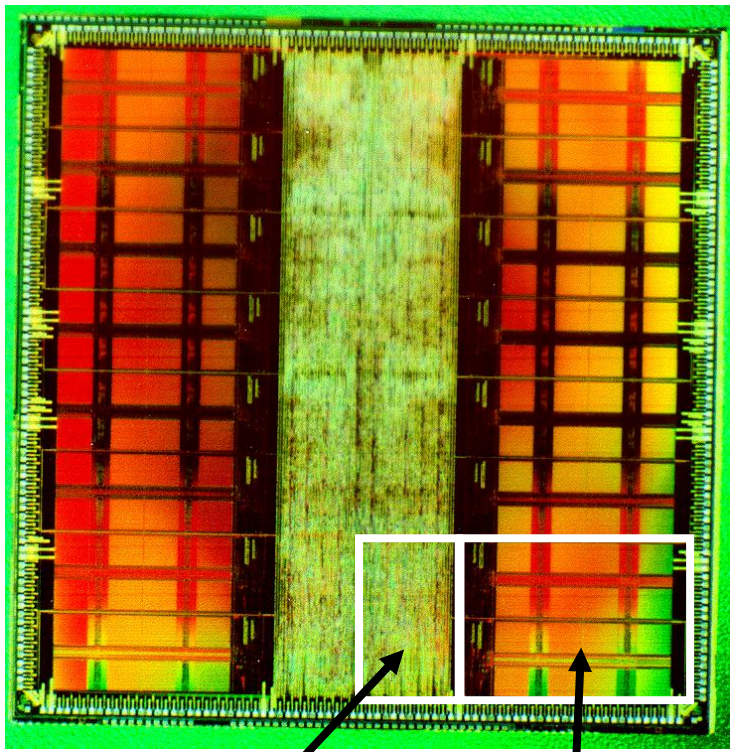
- Memory is as dumb today as it was 60 years ago
  - Even though its 90% of the silicon by area
  - Even though 45% of die IS NOT MEMORY
- Memory chip architectures focused on burst cache line bandwidth, not latency
- Performance dominated by # concurrent accesses from remote CPUs
- Modularity complicate reliable configurations
- Newest sense of “locality” still “processor centric”
  - Even though there are 10s-100s of memory chip and 1,000s to 100,000s of memory banks per locale

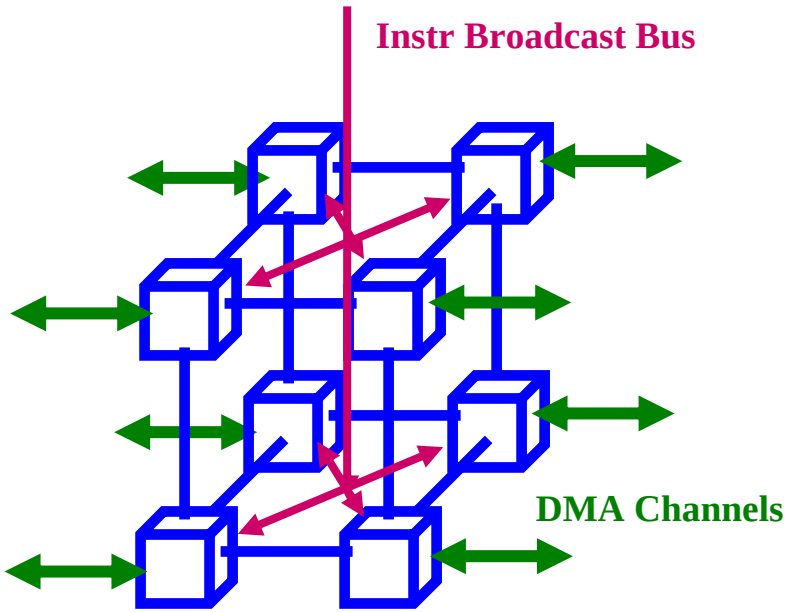
Q3: Most Exciting Development

- **Beginnings of *Relentless* Multi-Threading**
  - **First by Chip Multi-Processors (CMP)**
  - **Then limited 2 way**
  - **New combinations: eg. Niagara, Eldorado**
- **We can go further: Reduce thread state weight**
  - **Cheaper thread => more threads**
  - **More threads => more memory references**
  - **Lighter states => "Mobile threads"**
  - **Lighter states => Simple CPUs on memory die**

Q4: What Would I Have Done Differently?

- **Make the original EXECUBE "look like memory"**





3D Binary Hypercube
SIMD/MIMD on a chip

My Personal Vision

- Next Gen challenge: KD on massive dynamic graphs
  - Latency in realms of minimal reuse crucial
- Take Multi-Core to the Extreme
  - Memory chip architectures with multiple integrated simple processors next to individual memory banks
- Relentless multi-threading:
  - With thread state = cache line or smaller
- Switch from processor-centric to memory bank centric
- Let threads migrate: reducing latencies to 1-way